

Multidimensional Queries

Scalable Data Engineering

Motivation

Multidimensional Modeling

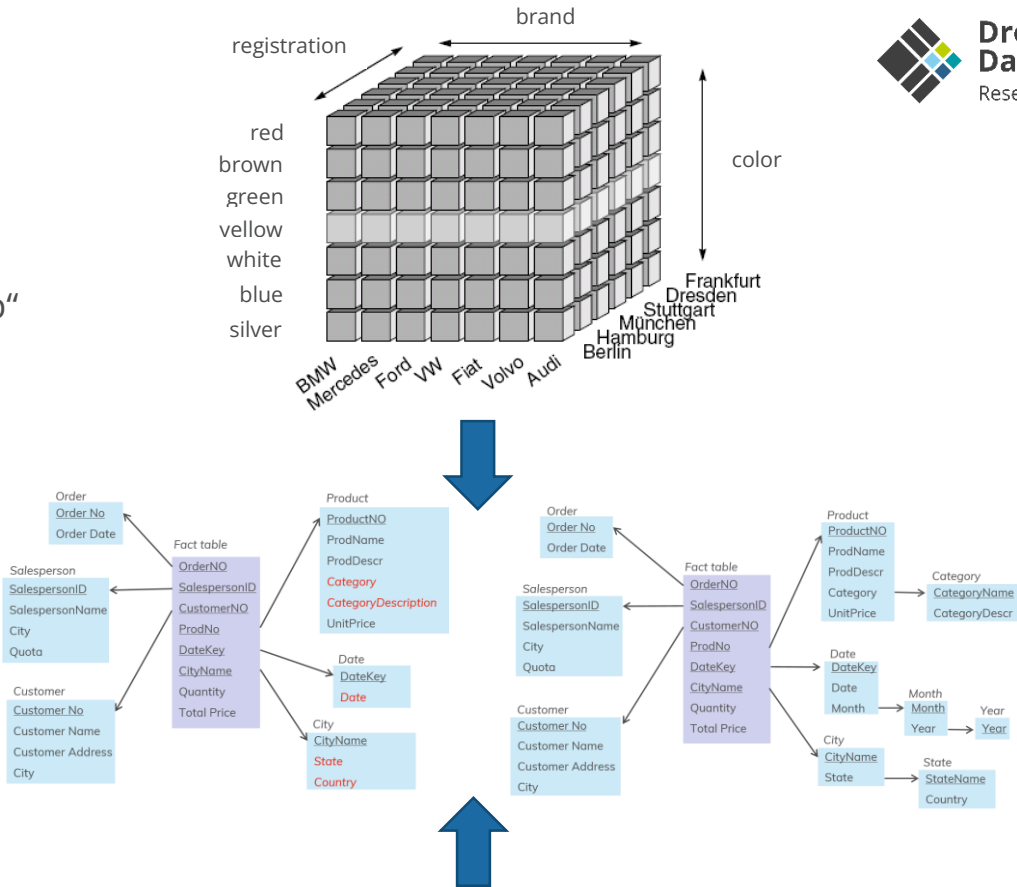
- Conceptual model data cube
- Multidimensional Operators „Roll Up“

Relational Mapping

- Star Schema
- Snowflake Schema

Query Processing

- SQL/OLAP extensions



```
SELECT Year, SUM(Total Price)
FROM Fact Table
GROUP BY Year
```

Recap SQL (only the querying part)

SQL Statement

- SELECT
 - select the columns that should appear in the result set, application of aggregation functions, definition of alternative column names
- FROM – select the source tables data should be read from, definition of JOINS
- WHERE – application of filters, alternative way of defining JOINS
- ORDER BY – defining the order of the result set

R:

year	quarter	sales
2020	3	10
2020	4	20
2021	1	10
2021	2	20
2021	3	30

```
select year, sales  
from R  
where year=2020
```

→

year	sales
2020	10
2020	20

Recap Grouping

Grouping (GROUP BY)

- Summarize tuples with equal values in grouping attributes into one new tuple
- Aggregation for non-grouping attributes
- Only grouping attributes, aggregates or constants are allowed in the select clause
- Most common aggregate functions: SUM(), MAX(), MIN(), COUNT(), AVG(), STDDEV()

year	quarter	sales
2020	3	10
2020	4	20
2021	1	10
2021	2	20
2021	3	30

```
select year, sum(sales)
from R
group by year
```

year	sum
2020	30
2021	60

Multidimensional Data with Standard SQL

Analysis

- By year and quarter
- By year
- Total

year	quarter	sales
2020	3	10
2020	4	20
2021	1	10
2021	2	20
2021	3	30



year	quarter	sum
-	-	90
2020	-	30
2021	-	60
2020	3	10
2020	4	20
2021	1	10
2021	2	20
2021	3	30

```
select year, quarter, sum(sales) from r group by year,quarter
union
select year, null, sum(sales) from r group by year
union
select null, null, sum(sales) from r
```

Agenda

Limits of Simple Grouping

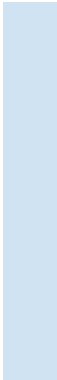
Overview SQL/OLAP Extension

Multiple Groupings

- Examples
- Implementation

Reporting Functions

Multi-Dimensional eXpressions - MDX



Overview SQL/OLAP Extension

Analysis using GROUP BY – Extensions

- Extensions of the GROUP-BY clause

GROUP BY ::=	<grouping column reference>
	<rollup list>
	<cube list>
	<grand total>
	<grouping sets list>
	<concatenated grouping>

} Extensions

- **GROUPING SETS** combine several groupings by different attributes and different grouping strategies
- **ROLLUP** create a hierarchical data cube using the listed attributes, aggregate along the dimensional hierarchies
- **CUBE** create all possible combinations of listed attributes

Scalar Functions

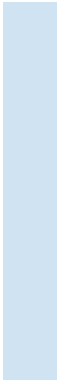
- LN (x) , EXP(x), POWER(x, n) , SQRT (x), FLOOR (x) , CEIL[ING] (x),
WIDTH_BUCKET (x, l, r, n)

Aggregate Functions

- STDDEV_POP(expr), STDDEV_SAMP(expr), VAR_POP(expr), VAR_SAMP(expr), COVAR_POP (expr1, expr2),
COVAR_SAMP (expr1, expr2), CORR (expr1, expr2)

Reporting Functions

- Windowed Table
- Window Function



Multiple Grouping

GROUPING SETS

- Grouping by several attributes simultaneously
- **GROUP BY GROUPING SETS** ((`<attribute-list>`), ...)
 - List of all desired attribute combinations
 - Empty set results in the superaggregate

year	quarter	sales
2020	3	10
2020	4	20
2021	1	10
2021	2	20
2021	3	30

```
select year, quarter, sum(sales)
from r
group by
    grouping sets
        ((), (year),
        (year, quarter))
```



year	quarter	sum
-	-	90
2021	1	10
2020	4	20
2020	3	10
2021	3	30
2021	2	20
2020	-	30
2021	-	60

Multiple Grouping

CUBE

- Calculation of all possible combinations of grouping attributes (power set)
- For n attributes there are 2^n combinations
- **GROUP BY CUBE** (<attribute-list>)

year	quarter	sales
2020	3	10
2020	4	20
2021	1	10
2021	2	20
2021	3	30

```
select year, quarter, sum(sales)
from r
group by
    cube (year, quarter)
```



year	quarter	sum
-	-	90
2021	1	10
2020	4	20
2020	3	10
2021	3	30
2021	2	20
2020	-	30
2021	-	60
-	3	40
-	4	20
-	2	20
-	1	10

ROLLUP

- Multidimensional grouping along dimensional hierarchies
- **GROUP BY ROLLUP** (<attribute-list>)
- **ROLLUP** ($A_1, A_2, \dots, A_N$) = **GROUPING SETS** ($()$, (A_1), (A_1, A_2), ..., ($A_1, A_2, \dots, A_N$))
- Applicable for functional dependencies between attributes
 - E.g., country determines continent:
ROLLUP (R_REGIONKEY, N_NATIONKEY)

year	quarter	sales
2020	3	10
2020	4	20
2021	1	10
2021	2	20
2021	3	30

```
select year, quarter, sum(sales)
from r
group by
    rollup (year, quarter)
```



year	quarter	sum
-	-	90
2021	1	10
2020	4	20
2020	3	10
2021	3	30
2021	2	20
2020	-	30
2021	-	60

GROUPING

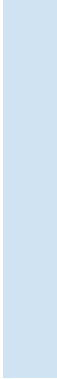
- Distinguish between actual and generated NULL values
- Applicable with ROLLUP and CUBE
- Returns
 - 0: attribute is part of the current grouping
 - 1: attribute is NOT part of the current grouping

year	quarter	sales
2020	3	10
2020	4	20
2021	1	10
2021	2	20
2021	3	30
2019	-	33

```
select year, quarter,  
       sum(sales),  
       grouping (quarter)  
from s  
group by  
       rollup (year, quarter)
```



year	quarter	sum	grp
-	-	123	1
2021	1	10	0
2020	4	20	0
2020	3	10	0
2021	3	30	0
2019	-	33	0
2021	2	20	0
2020	-	30	1
2019	-	33	1
2021	-	60	1



Implementation of Multiple Groupings

GROUP BY – Implementation

Dictionary/Map approach

- Key entry for every group, value collects the aggregate
- Problem: Dictionary might become very large

2019	20
2020	28



year	quarter	sales
2020	3	10
2020	4	20
2021	1	10



Sorting Approach

- Sort data by the grouping attribute
- All entries of the same group can be processed consecutively
- No hot intermediate results

year	quarter	sales
2020	3	10
2020	4	20
2021	1	10
2021	2	20
2021	3	30



2020	#
------	---

...

year	quarter	sales
2020	3	10
2020	4	20
2021	1	10
2021	2	20
2021	3	30



2021	#
------	---

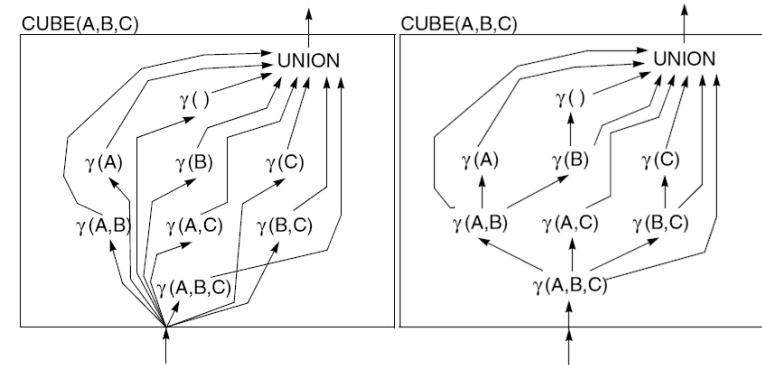
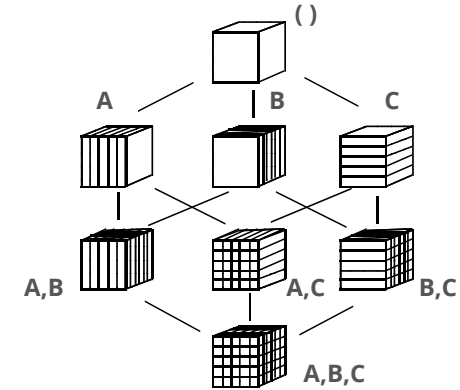
CUBE – Implementation

Problem

- CUBE() results in 2^n attribute combinations
- Example: Three attributes $\rightarrow$ eight grouping operations with preceding sort operations

Naive Approaches

- Left: 2^n query executions and a succeeding union
- Right: Improvement by taking advantage of direct derivability
 - Pro: Stepwise reduction of cardinality („subset stacking“)
 - Choose smallest direct predecessor („smallest parent“), e.g. derive $\gamma()$ from $\gamma(B)$ instead of $\gamma(A)$ or $\gamma(C)$



„Shared sort“

- Choose an ordering that supports higher groupings
- Grouping path: $(ABCD) \rightarrow (ABC) \rightarrow (AB) \rightarrow (A) \rightarrow ()$ (order by A, B, C, D only once)
- Example
 - Data set is ordered by ABC
 - $\gamma(AB)$ can be calculated right away
 - $\gamma(BC)$ requires order by BCA
- Problem: data set has to be reordered to address different groupings

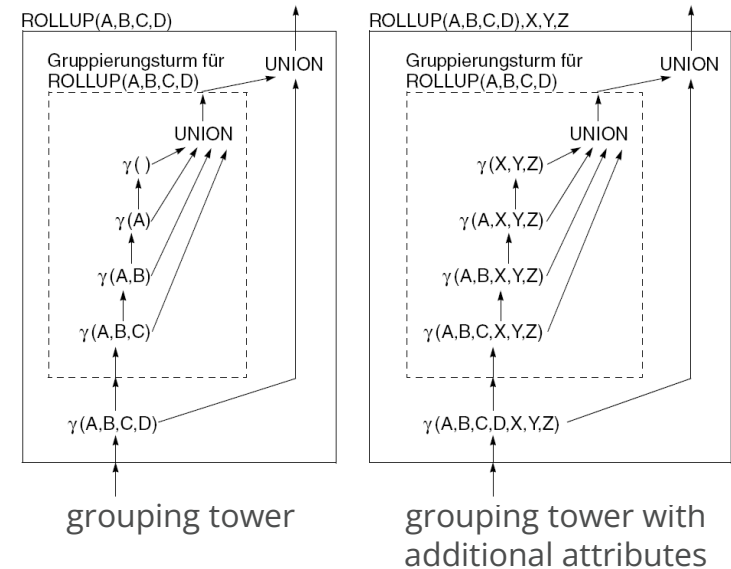
Problem

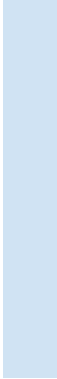
- Conflicting strategies „smallest parent“ and „shared sort“
- Example
 - (AB) could be calculated from (BDA) which might be smaller than (ABC) , but the data set is already ordered by (ABC)

ROLLUP – Implementation

Sorting approach

- ROLLUP can be implemented by only one sort operation on the finest granularity
- Left: sort once by (ABC), when (ABCD) is the base granularity
- Right: extended example with additional grouping attributes that are not part of the grouping operation





Reporting Functions

Overview Reporting Functions

Basics

- Scalar functions (per tuple): e.g. $+$ $-$ $*$ $/$
- Grouping: **GROUP BY**
 - Summarize tuples with equal values in grouping attributes
- Aggregate functions: e.g. **SUM**, **COUNT**, **AVG**
 - Calculation of one value from several tuples
- Grouping and aggregates are usually used together

OLAP functions (analytic functions a.k.a. reporting functions)

- Extensions of aggregate functions
- Separations of grouping and aggregation
 - Further aggregation on **query result**
 - Not aggregating the result itself
- Core construct: **OVER** clause

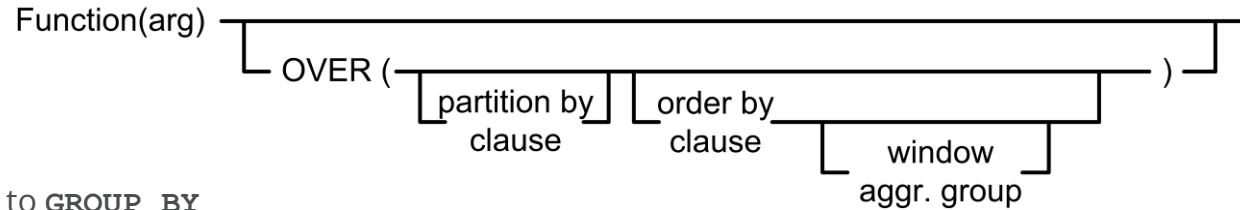
Overview Reporting Functions

Syntax

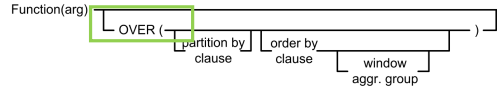
- **PARTITION BY** and **ORDER BY** are optional
- **WINDOW** specification is possible

Clause elements

- **PARTITION BY**
 - Partitioning of a relation similar to **GROUP BY**
- **ORDER BY**
 - Ordering of tuples in a partition
- **WINDOW**
 - Position based (**ROWS**): relative to current row (preceding and following rows)
 - Range based (**RANGE**): values within the same partition
- Aggregate function



OVER()



OVER clause

- Aggregation without grouping
- **OVER** () aggregate all tuples

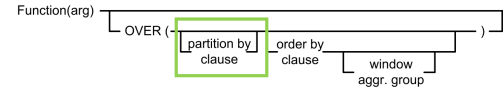
year	quarter	sales
2020	3	10
2020	4	20
2021	1	10
2021	2	20
2021	3	30

```
select year, quarter, sales,  
       sum(sales) over()  
from r
```



year	quarter	sales	sum
2020	3	10	90
2020	4	20	90
2021	1	10	90
2021	2	20	90
2021	3	30	90

Partitioning



Partitioning of attributes

- Selection of partitioning attributes, similar to **GROUP BY**
- **PARTITION BY** <attribute-list>

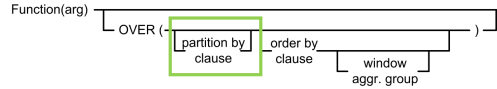
year	quarter	sales
2020	3	10
2020	4	20
2021	1	10
2021	2	20
2021	3	30

```
select year, quarter, sales,  
       sum(sales) over(partition by  
                        year)  
from r
```



year	quarter	sales	sum
2020	3	10	30
2020	4	20	30
2021	1	10	60
2021	2	20	60
2021	3	30	60

Partitioning



Partitioning of attributes

- Selection of partitioning attributes, similar to **GROUP BY**
- **PARTITION BY** <attribute-list>
- Can be used for further calculations, e.g. quarterly sales and their share in annual sales (cast necessary!)

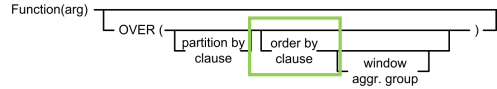
year	quarter	sales
2020	3	10
2020	4	20
2021	1	10
2021	2	20
2021	3	30

```
select year, quarter, sales,  
       sales/sum(sales) over(  
         partition by year)::numeric  
from r
```



year	quarter	sales	sum
2020	3	10	0.333
2020	4	20	0.667
2021	1	10	0.167
2021	2	20	0.333
2021	3	30	0.500

Partition ordering



Attribute local ordering

- Ordering of tuples within one attribute partition, similar to **order by**
- **ORDER BY** <attribute-list> [direction]
- Not only change in order, order attributes are added to the partitioning!!!

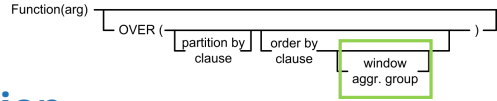
year	quarter	sales
2020	3	10
2020	4	20
2021	1	10
2021	2	20
2021	3	30

```
select year, quarter, sales,
sum(sales) over(partition by
year order by quarter desc)
from r
```



year	quarter	sales	sum
2020	4	20	20
2020	3	10	30
2021	3	30	30
2021	2	20	50
2021	1	10	60

Window definition

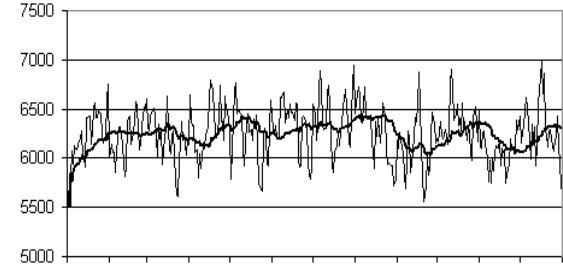


Position (and partition) based window definition

- ROWS BETWEEN *m* PRECEDING AND *n* FOLLOWING
- ROWS BETWEEN *m* PRECEDING AND CURRENT ROW
- RANGE CURRENT ROW

Example

- Moving average, window size 3



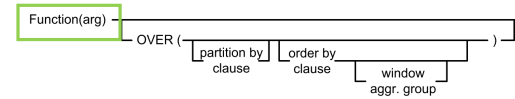
year	quarter	sales
2020	3	10
2020	4	20
2021	1	10
2021	2	20
2021	3	30

```
select year, quarter, sales,  
       sum(sales) over(partition by  
                        year order by quarter rows  
                        between 1 preceding and 1  
                        following)  
from r
```



year	quarter	sales	sum
2020	3	10	30
2020	4	20	30
2021	1	10	30
2021	2	20	60
2021	3	30	50

Aggregate Function



Window Function

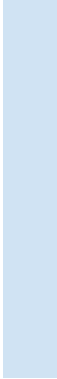
- Operates within one partition
- Output only represents the tuples within one partition (or maybe total)
- Five windowed table functions added for reporting purpose
 - `RANK ()`, `DENSE_RANK ()`, `PERCENT_RANK ()`, `CUME_DIST ()`, `ROW_NUMBER ()`

year	quarter	sales
2020	3	10
2020	4	20
2021	1	10
2021	2	20
2021	3	30

```
select year, quarter, sales,  
rank() over(order by sales desc),  
cume_dist() over(order by sales  
asc)  
from r
```



year	quarter	sales	rank	cume
2020	3	10	4	0.4
2021	1	10	4	0.4
2020	4	20	2	0.8
2021	2	20	2	0.8
2021	3	30	1	1



MDX

MDX (= Multi-Dimensional eXpressions)

- Microsoft Analysis Services (OLAP services)
- Developed by Microsoft, now industry standard, although not supported by all data bases (e.g. PostgreSQL ☹)

Structure of MDX Queries

- Partly based on SQL syntax
- Partly related graphical query specification

Language Reference

- <http://technet.microsoft.com/de-de/library/ms145506.aspx>

Tutorial

- <http://www.mdx-hilfe.de/>
 - Only available in German, might be helpful anyway

SELECT (axis definition)

- Set of elements (classification nodes) **ON COLUMNS**
- Set of elements (classification nodes) **ON ROWS**
- Also: **ON PAGES, SECTIONS, CHAPTERS, ...** for more than two dimensions
- In general: **ON AXIS(i)**

FROM (source specification)

- Multidimensional join operation of data cubes
- Join-compatibility: data cube must have at least one common dimension

WHERE (restrictions/filters)

- Restrictions regarding the domain of the data cube

Measures

- Elements of the always available dimension “Measures”
- Standard aggregate functions are applicable (SUM, MIN, MAX, COUNT)

Structure of MDX Queries

```
SELECT {Germany,  
        Bavaria,  
        Frankfurt, Erlangen}  
        ON COLUMNS,  
{Qtr1.CHILDREN,  
  Qtr2, Qtr3,  
  Qtr4.CHILDREN} ON ROWS  
  
FROM SalesCube  
  
WHERE (Measures.Sales,  
        Time.[2002],  
        Products.[All Products]);
```

} axis definition

} source specification

} measures

} restrictions/filters

Enumeration

- Enumeration of elements of different hierarchy levels
- Example (nation dimension)
`{USA, California, [San Francisco], [San Jose], Germany, Erlangen, Saxony}`

Element Expressions

- `USA.CHILDREN: {CA, TX, ....}` all direct children of a certain entry
- `CA.PARENT: USA` the parent entry of the next higher hierarchy level
- `Products.ALL` root node of the hierarchy
- `Time.Quarter.MEMBERS` list all members of one node

Functional Production of Sets

- `DESCENDANTS(USA, Cities)` all children on a certain hierarchy level
- `GENERATE ({USA, France}, DESCENDANTS(Geography.CURRENT, Cities))` generate a list of all cities in USA and France, country and city are levels in the Geography hierarchy

Set Expressions in MDX

Nesting of Sets

- Combination of sets for axis definition

```
SELECT CROSSJOIN({Germany, Bavaria, Frankfurt, Erlangen}  
                  {FurnitureBox, [H&M-Furniture]}) ON COLUMNS,  
              {Qtr1.CHILDREN, Qtr2, Qtr3, Qtr4.CHILDREN} ON ROWS  
FROM SalesCube  
WHERE (Measures.Sales, [2020], Products.[All Products]);
```

	Germany		Bavaria		Frankfurt		Erlangen	
	Furniture Box	H&M- Furniture	Furniture Box	H&M- Furniture	Furniture Box	H&M- Furniture	Furniture Box	H&M- Furniture
Jan 2020								
Feb 2020								
Mar 2020								
Qtr2 2020								
Qtr3 2020								
Oct 2020								
Nov 2020								
Dec 2020								

sales values

Relative Selection

- Exploit order in dimensional structure
 - `Time.[2002].LastChild`: Fourth Quarter 2002
 - `[2002].NextMember`: {[2003]}
 - `[2002].[Qtr4].Nov.Lead(2)`: January 2003
 - `[1992]:[2002]:[1992], ..., [2002]`

Working the Dimensions

- Access hierarchical meta information
- `Germany.LEVEL`: Country
- `Time.LEVELS(1)`: Year (starting with the highest/most coarse grained level of a hierarchy)

Parentheses

- `{}`: sets, {Mike, John}
- `[]`: special characters, spaces and non-numeric interpretation of numbers, [Nürnberg], [San Jose], [2002]
- `()`: tuples (Sales, [2002]) in WHERE clause

Summary

Limits of Simple Grouping

Overview SQL/OLAP Extension

Multiple Groupings

- Examples
- Implementation

Reporting Functions

Multi-Dimensional eXpressions - MDX

Exercise

PostgreSQL

- <https://www.postgresql.org/download/>

PG-Admin

- <https://www.pgadmin.org/download/>

TPCH

- <https://cloudstore.zih.tu-dresden.de/index.php/s/6BqBj3CdFZ8ddY5>